

# Art and Sam Integration

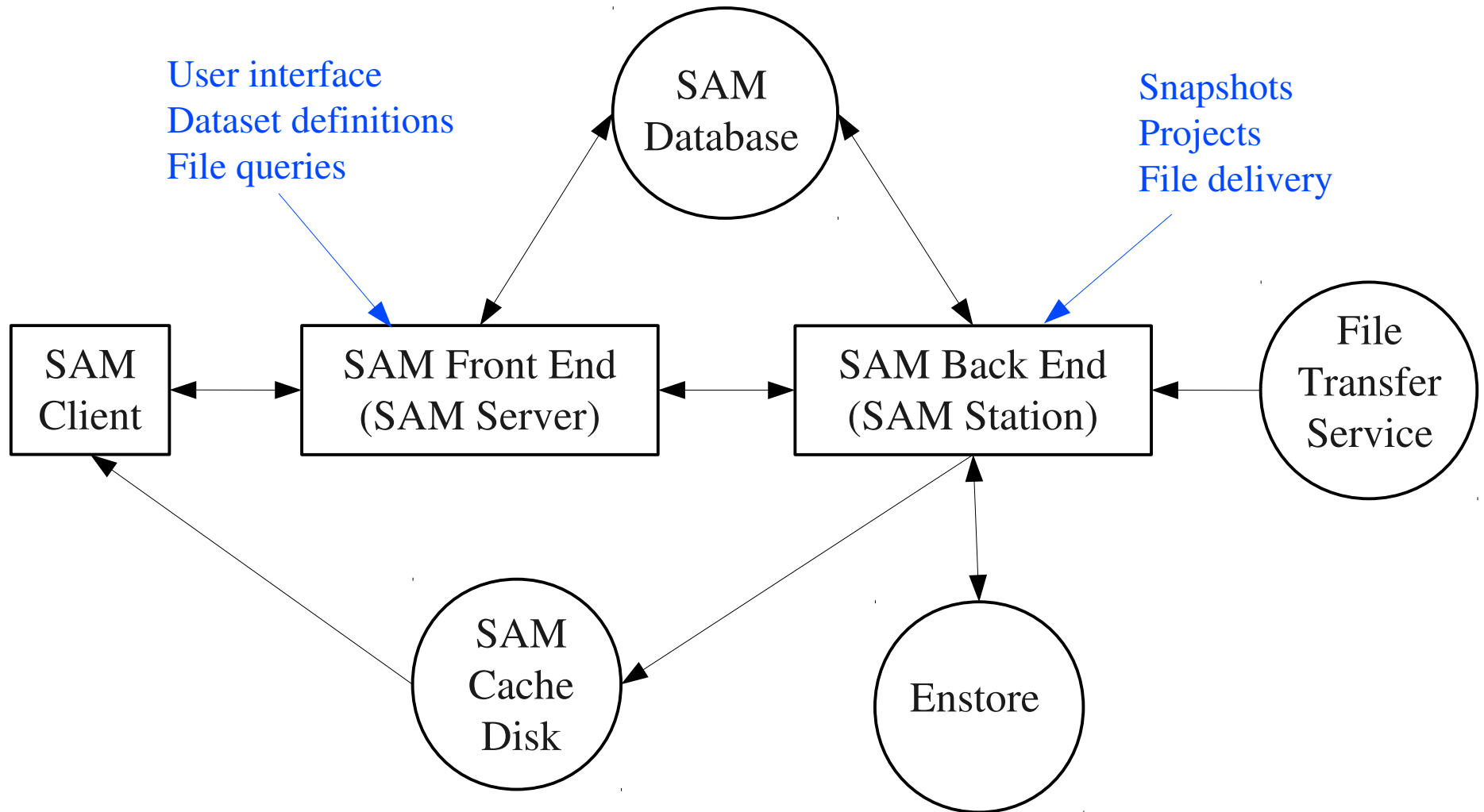
General Larsoft Meeting  
Oct. 2, 2013

H. Greenlee

# Introduction

- The last time I talked about this topic in a larsoft meeting was the May 8, 2013 general larsoft meeting (talk in redmine and indico).
  - In this talk, I will summarize what I said then, and emphasize what has changed, and what progress has been made.

# Data Handling Overview



# Sam Clients

- All sam clients have in common that they send requests to the samweb http server (<http://samweb.fnal.gov:8480/sam/uboone/api>)
- Samweb (setup sam\_web\_client).
  - Line mode client (samweb -e uboone <subcommand> ...).
  - Python client (import samweb\_cli).
- Ifdh client tools (setup ifdhc).
  - Line mode client (ifdh <subcommand>).
  - Python client (import ifdh).
  - C++ client (class ifdh, not art-specific).
- Art client (setup ifdh\_art).
  - Wraps ifdhc c++ sam client as art service (IFDH service), and provides sam-capable instances of file delivery and file transfer services.

# Art + Sam Use Cases

- Sam output (generate sam metadata).
  - This use case does not actually require sending requests to the sam server. You just have to know what metadata you want to associate with each output file.
  - This use case does not include declaring files to sam or uploading files to enstore. These things are optional and external to art program.
- Sam input.
  - This use case requires that the art program send “fetch next file” and “release file” type requests to the sam server.
  - Other communication with the server to initialize and deinitialize sam project is required, but is external to art program.

# Sam Output Art Services and Modules

- FileCatalogMetadata service (art).
  - Defines basic metadata.
- RootOutput module (art).
  - Defines basic metadata.
- FileCatalogMetadataExtras service (larsoft/Utilities).
  - Does stuff that art should do, but doesn't.
    - Arbitrary per-job metaedata (name, value).
    - Standard sam per-file metadata.
      - First event, last event, number of events.
      - Time stamps.
      - Run number, subrun number.
      - Parent files.
    - Copying arbitrary metadata from input file to output file.
    - Generating unique output file names from a template.

# SAM Input Art Services and Modules

- CatalogInterface service (art). Pure virtual class. Derived classes:
  - TrivialFileDelivery service (art). Supports files and file lists.
  - IFCatalogInterface service (ifdh\_art). Supports sam/ifdh.
- FileTransfer service (art). Pure virtual class. Derived classes:
  - TrivialFileTransfer service (art). Supports files and file lists.
  - IFFileTransfer service (ifdh\_art). Supports sam/ifdh.
- IFDH service (ifdh\_art). Full C++ samweb client.
- RootInput module.

# IFDH\_ART

- Sam input support mostly provided by art services that live in ups product ifdh\_art.
- All of the ifdh\_art art services mentioned on the previous slide work, can be used today.
- Currently, you need to setup ifdh\_art by hand.
  - `setup ifdh_art v1_2_1 -q debug:e2:nu`
- Above setup should be included in standard larsoft setup.
  - Doesn't need to wait for larsoft reconfiguration.



# Configuring Sam Services and Modules

- Sam input and sam output services and modules are completely independent. You can use sam input and sam output separately or together.
- In general, there is no reason to interact with sam services in user code. You just need to adjust your fcl job configuration to enable sam input and output.

# Example Sam Output Job FCL File

```
services:
{
  FileCatalogMetadata:
  {
    applicationFamily:    "art",
    applicationVersion:   "S2013.06.25",
    fileType:             "mc"
  }
  user:
  {
    FileCatalogMetadataExtras:
    {
      Metadata: [ "group",      "uboone",
                  "fileFormat", "root",
                  "fclName",    "standard_reco_uboone.fcl",
                  "fclVersion", "v1_5" ],
      GeneratePerFileMetadata: true
      CopyMetadataAttributes:  [ "fileType", "runType" ]
    }
  }
}

outputs:
{
  out1:
  {
    module_type: RootOutput
    fileName:    "standard_reco_uboone.root"
    dataTier:    "reconstructed"
  }
}
```

# Sam Input: Project Life Cycle

1) Generate unique project name.

- Can be done in submit script. Name can be anything. There is a samweb helper command..

2) Start project.

- Can be done in various places. I prefer having a separate batch job.

3) Start consumer process.

- Should be done in batch worker script, before starting art program.
- There can be many workers, and many consumer processes, in a sam project.

4) File loop.

- a) Get location (uri) of next file.
- b) Copy file to scratch disk.
- c) Process file.
- d) Release file.
- e) Delete file from scratch disk.

These steps take place  
inside the art program.

5) Stop project.

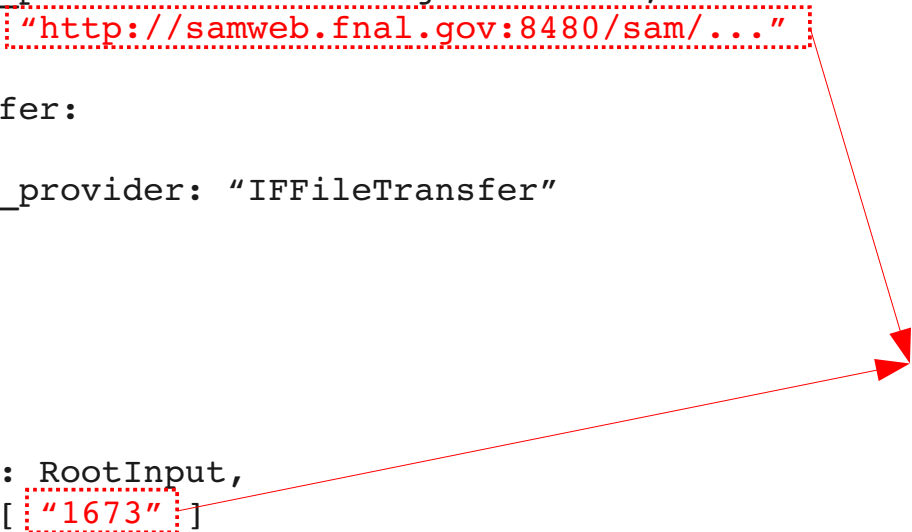
- Should be separate batch job.

# Example Sam Input Job FCL File

```
services:
{
  user:
  {
    IFDH:
    {
      IFDH_BASE_URI: "http://samweb.fnal.gov:8480/sam/uboone/api"
    }
    CatalogInterface:
    {
      service_provider: "IFCatalogInterface",
      webURI: "http://samweb.fnal.gov:8480/sam/..."
    }
    FileTransfer:
    {
      service_provider: "IFFileTransfer"
    }
  }
}

source:
{
  module_type: RootInput,
  fileNames: [ "1673" ]
}
```

Project url and  
consumer process id  
are only known on  
batch worker



# Worker Level Configuration

- The project url and consumer process id fcl parameters must be set inside the batch worker.
  - There are command line overrides, but they are buggy (or featury) in the current version of art.
  - I prefer to set all sam fcl parameters in the fcl file. Easiest way to do this is to make a wrapper fcl file. Example:

```
#include "myjob.fcl"
```

```
services.user.CatalogInterface.webURI: "http://samweb.fnal.gov:8480/sam/..."  
source.fileNames: [ "2932" ]
```

# Using DAG to Serialize Start and Stop Project Batch Jobs

- I told you three slides ago that starting and stopping the project should be done in separate batch jobs.
- You can use the DAG (directed acyclic graph) feature of condor/jobsub to serialize start project, worker, and stop project batch jobs.
- Submit jobs using command `dagNabbit.py myjob.dag`.
  - Script `dagNabbit.py` is included in `jobsub_tools` (front end for jobsub).
  - Example `.dag` file:

```
<serial>
jobsub -n -g ... condor_start_project.sh ...
jobsub -n -g -N 100 ... condor_lar.sh ...
jobsub -n -g ... condor_end_project.sh ...
</serial>
```

- Unfortunately, `dagNabbit.py` is broken if your login shell on `gpsn01` is `(t)csh`. Hopefully should be fixed soon...

# Using SAM Interactively

- Reading files from sam (using art programs or scripts) works on any node. But there are a couple of gotchas for using sam in an interactive environment.
  - The “scratch disk” directory used by ifdh is defined by environment variable **TMPDIR**. For interactive use, you must define this environment variable by hand, or you will fill up /var/tmp on the gpvm nodes.
    - In a batch environment, this environment variable is defined for you by the batch system.
  - Interactively, use **kinit + get-cert** to authenticate yourself to the samweb server for commands requiring authentication.
    - In a batch environment, you automatically have the right credentials.

# Summary

- Reviewed how to generate sam metadata in output files.
  - See May 8 talk for more details.
  - FileCatalogMetadataExtras has some new features since May 8 talk.
- Sam input support in art is mainly provided by services that live in ifdh\_art ups product.
  - Sam project life cycle.
  - Art job configuration.
  - How to use dag to submit sam project batch jobs.